＜論 説＞

AI手法（ガウス過程）を用いた予測――数値篇

荒 木 勝 啓

目　　次

Ⅰ　ガウス過程回帰におけるモデル選択

(i)　カーネル関数と共分散行列の計算方法

ガウス過程による回帰予測の公式は拙著「AI手法（ガウス過程）を用いた予測–理論篇」[1] で示した通り次の３つの式に集約される。

$$p(\boldsymbol{y}_*|\boldsymbol{y}) = N(\boldsymbol{y}_* \mid \boldsymbol{\mu}_{\boldsymbol{y}_*|\boldsymbol{y}}, \Sigma_{\boldsymbol{y}_*|\boldsymbol{y}}) \tag{1}$$

$$\boldsymbol{\mu}_{\boldsymbol{y}_*|\boldsymbol{y}} = \boldsymbol{\mu}_* + \boldsymbol{K}(\boldsymbol{X}_*, \boldsymbol{X})\left[\boldsymbol{K}(\boldsymbol{X}, \boldsymbol{X}) + \sigma_y^2 \boldsymbol{I}\right]^{-1} \boldsymbol{y} \tag{2}$$

$$\Sigma_{\boldsymbol{y}_*|\boldsymbol{y}} = \boldsymbol{K}(\boldsymbol{X}_*, \boldsymbol{X}_*) - \boldsymbol{K}(\boldsymbol{X}_*, \boldsymbol{X}) \left[\boldsymbol{K}(\boldsymbol{X}, \boldsymbol{X}) + \sigma_y^2 \boldsymbol{I}\right]^{-1} \boldsymbol{K}(\boldsymbol{X}, \boldsymbol{X}_*) \tag{3}$$

ただし $\boldsymbol{X}$ は n 個の訓練データ $x_i (i = 1,2,\cdots,n)$ から成る $m \times n$ 行列

$$\boldsymbol{X} = [\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_n] = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1n} \\ x_{21} & x_{22} & \cdots & x_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ x_{m1} & x_{m2} & \cdots & x_{mn} \end{bmatrix} \quad (m \times n \text{行列}) \tag{4}$$

であり、$\boldsymbol{y} = (y_1, y_2, \cdots y_n)^T$ はそれぞれの訓練に対する「解答」すなわち教師データである観測値である。ただし m は説明変数の数、n はデータの組の個数である。$\boldsymbol{X}_*$ はテスト・データ $\boldsymbol{X}_* = \left[\boldsymbol{x}_1^*, \boldsymbol{x}_2^*, \cdots, \boldsymbol{x}_h^*\right]$ で、このデータの組が追加されることにより予測される値のベクトル、すなわちテストに対する予測的解答が、(2) と (3) 式で与えられている。$\boldsymbol{\mu}_{\boldsymbol{y}_*|\boldsymbol{y}}$ は通常の記法を使うと $\boldsymbol{\mu}_{\boldsymbol{y}_*|\boldsymbol{y}} = \hat{\boldsymbol{y}}$（ただし上付きハットは予測を表す）と表すことができる。

本稿では、時間を訓練データ、時系列データを教師データとして、将来の時系列経路を予測するというシミュレーションを行う。その文脈で (4) と $\boldsymbol{X}_* = \left[\boldsymbol{x}_1^*, \boldsymbol{x}_2^*, \cdots, \boldsymbol{x}_h^*\right]$ を書き直すと、訓練データ (4) は１変数 n 個の $1 \times n$ ベクトル

$$\boldsymbol{X} = [\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_n] = \left[x_{11} \quad x_{12} \quad \cdots \quad x_{1n}\right] = \left[t_1 \quad t_2 \quad \cdots \quad t_n\right] = \left[1 \quad 2 \quad \cdots \quad n\right]$$

追加されるテスト・データは $n+1$ から先 $n+h$ までの

$$\boldsymbol{X}_* = [n+1, \quad n+2, \quad \cdots, \quad n+h]$$

$1 \times h$ ベクトルということになる。

1　文献[9].

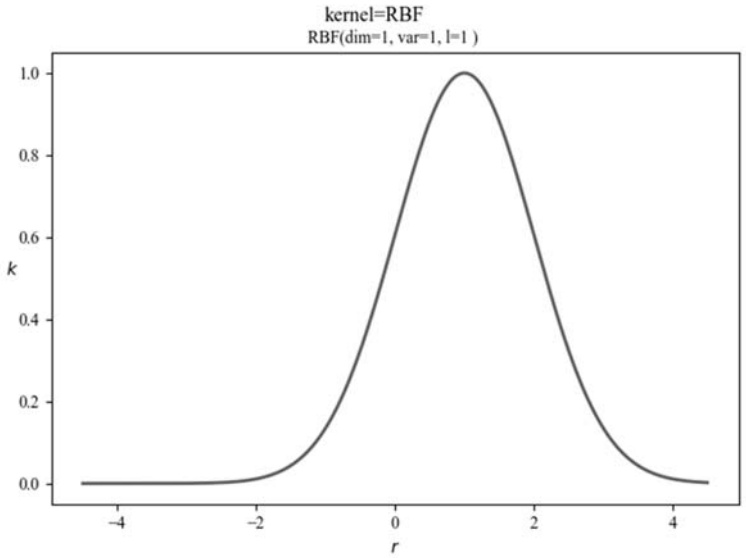

図 1　RBF カーネル関数

カーネル関数$k(x_i, x_j)$と共分散行列$\boldsymbol{K}(\boldsymbol{X}, \boldsymbol{X})$あるいは$\boldsymbol{K}(\boldsymbol{X}_*, \boldsymbol{X})$の直感的理解には少し説明を要する。典型的例として RBF(Radial Basis Function):

$$k\left(\boldsymbol{x}_i,\ \boldsymbol{x}_j\right)=\sigma^2\exp\left(\frac{\left(\boldsymbol{x}_i-\boldsymbol{x}_j\right)^T\left(\boldsymbol{x}_i-\boldsymbol{x}_j\right)}{2\lambda^2}\right) \quad (5)$$

を取り上げる。まず説明変数が 1 変数xの場合例えば上の例のように$x=t$とすればn個のデータは$(t_1 \quad t_2 \quad \cdots \quad t_n)$であり (5) 式はスカラー数$t_i$と$t_j$により

$$k\left(t_i,\ t_j\right)=\sigma^2\exp\left(\frac{\left(t_i-t_j\right)^2}{2\lambda^2}\right) \quad (6)$$

となる。$\sigma^2=1, \lambda=1, i=j, t_i=i$ならば (6) の値は

$$k(1,\ 1)=\exp\left(-\frac{(1-1)^2}{2}\right)=\exp\left(-\frac{0}{2}\right)=\exp(0)=1$$

同様に

$$k(1,\ 2)=k(2,\ 1)=\exp\left(-\frac{(1-2)^2}{2}\right)=\exp\left(-\frac{1}{2}\right)=0.6065\cdots$$

などと計算できる。t_iを 1 としてこの点を中心にt_jを連続的に動かしていけば RBF カーネル関数は図 1 に示すような 1 を中心とした左右対称な関数形となる。

こうしたカーネル関数から共分散行列を構築すると例えば$n=2$の場合、1 変数$(m=1)$、2 データ$(n=2)$、は横ベクトル$(\boldsymbol{X}=(1, 2))$、$\boldsymbol{K}(\boldsymbol{X}, \boldsymbol{X})$は$n\times n=2\times 2$行列として

$$\boldsymbol{K}(\boldsymbol{X},\boldsymbol{X})=\begin{bmatrix}k(t_1,t_1) & k(t_1,t_2)\\ k(t_2,t_1) & k(t_2,t_2)\end{bmatrix}=\begin{bmatrix}k(1,1) & k(1,2)\\ k(2,1) & k(2,2)\end{bmatrix}=\begin{bmatrix}1 & 0.6065\cdots\\ 0.6065\cdots & 1\end{bmatrix}$$

のような対称行列となる。ここで注意すべきは例えば新たに$h=3$個のテスト・データ$\boldsymbol{X}_*=(t_3,t_4,t_5)=(3,4,5)$が追加された場合の$\boldsymbol{K}(\boldsymbol{X}, \boldsymbol{X}_*)$は

$$\boldsymbol{K}(\boldsymbol{X},\boldsymbol{X}_*)=\begin{bmatrix}k(1,3) & k(1,4) & k(1,5)\\ k(2,3) & k(2,4) & k(2,3)\end{bmatrix}$$

となり、$n\times h=2\times 3$の非対称行列となることである。(3) 式の行列の行数と列数を追っていくと、右辺第 1 項は 、第 2 項の行列掛け算は$(h\times n)\sim(n\times n)\sim(n\times h)=h\times h$であるから右辺の途中の計算中にこのような非対称行列が現れても、左辺は$h\times h$の対称行列すなわち分散共分

散行列として定義されることが分かる。

説明変数が m 個ある (4) 式の $\boldsymbol{x}$ の場合、データの組が n 個あると共分散行列は

$$\boldsymbol{K}(\boldsymbol{X},\boldsymbol{X})=\begin{bmatrix} k(\boldsymbol{x}_1,\boldsymbol{x}_1) & k(\boldsymbol{x}_1,\boldsymbol{x}_2) & \cdots & k(\boldsymbol{x}_1,\boldsymbol{x}_n) \\ k(\boldsymbol{x}_2,\boldsymbol{x}_1) & k(\boldsymbol{x}_2,\boldsymbol{x}_2) & \cdots & k(\boldsymbol{x}_2,\boldsymbol{x}_n) \\ \vdots & \vdots & \cdots & \vdots \\ k(\boldsymbol{x}_n,\boldsymbol{x}_1) & k(\boldsymbol{x}_n,\boldsymbol{x}_2) & \cdots & k(\boldsymbol{x}_n,\boldsymbol{x}_n) \end{bmatrix}$$

と定義される。カーネル関数の計算は、変数がベクトルであっても、(5) 式を例にとると、パラメーターの他は 2 点間のユークリッド距離 $r=\sqrt{(\boldsymbol{x}_i-\boldsymbol{x}_j)^T(\boldsymbol{x}_i-\boldsymbol{x}_j)}$ にのみ依存するので、例えば

$$k(\boldsymbol{x}_1,\boldsymbol{x}_2)=k(r)=\sigma^2\exp\left\{-\frac{r^2}{2\lambda^2}\right\}$$

$$=\sigma^2\exp\left\{-\frac{(x_{11}-x_{12})^2+(x_{21}-x_{22})^2+\cdots+(x_{m1}-x_{m2})^2}{2\lambda^2}\right\}$$

のようにスカラー値の四則演算によりスカラー値として求められる。

ガウス過程とパラメトリック予測を比較すると、パラメトリック推定の計算上難しい点があるとすれば、前掲著 (9) や (10) 式に現れる逆行列の計算であるものの、これらは説明変数の個数 m による小規模の $m\times m$ 逆行列の計算であるのに対し、ガウス過程では (2) (3) 式に現れる $\left[\boldsymbol{K}(\boldsymbol{X},\boldsymbol{X})+\sigma_y^2\boldsymbol{I}\right]^{-1}$ という $n\times n$ の大規模な逆行列を計算しなければならないという点である。m は説明変数の個数であるので比較的小さいが、n はデータの個数なので通常は非常に大きい。例えば日経平均の終値データは 1 年分で約 250 ある。10 分ごとの為替データだと 1 か月 25 日として 1 か月分で 3500 にもなる。従ってガウス過程の応用で最大の鍵となるのが、高いオーダーの逆行列計算をいかに効率的・高速に行えるかであり、これはこれで一大研究分野を成している。幸い GPy モジュール [2] ではこの問題は実用的レベルまで解決されているので、本稿では逆行列計算に使われる行列の分解公式などの解説を省略する。

(ii)　カーネル関数の組み合わせによるモデル設計

一般に回帰分析ではカーネル関数の形状はガウス過程回帰の結果としての曲線を局所的に近似している。従って各カーネルを足し算掛け算で組み合わせることにより、元データの形に近い形状を生み出すことができる。この組み合わせをモデル設計と呼ぶ。モデル設計が適切であればあるほどガウス過程回帰による予測は正確になると期待される。

以下 $\boldsymbol{X}=(-2{,}7,\ -1{,}8{,}5{,}14)$ を例にこのことを検討してみよう。

(a)　線形カーネル

一般に説明変数が m 個のベクトルの第 i 番目のデータと第 j 番目のデータを

$$\boldsymbol{x}_i=\begin{bmatrix} x_{1i} \\ \vdots \\ x_{mi} \end{bmatrix},\ \boldsymbol{x}_j=\begin{bmatrix} x_{1j} \\ \vdots \\ x_{mj} \end{bmatrix} \tag{7}$$

として

$$k(\boldsymbol{x}_i,\boldsymbol{x}_j)=\sigma_1^2x_{1i}x_{1j}+\sigma_2^2x_{2i}x_{2j}+\cdots+\sigma_m^2x_{mi}x_{mj}$$
$$=\sum_{s=1}^{m}\sigma_s^2x_{si}x_{sj} \tag{8}$$

と定義されるのが GPy.kern[3] における線形カーネルである [4]。特に説明変数が 1 個 ($\boldsymbol{x}=x$) の時線形カーネルは、

$$k(x_i,x_j)=\sigma^2x_ix_j \tag{9}$$

という簡単な定義となる。(9) 式は $r=x_ix_j$ と置けば図 2 に示すように横軸を r 傾きを σ^2 とする原点を通る直線で表される。ただし σ^2 を 0.4 と置いた。このカーネルを元にガウス過程回帰 (optimize 前) を求めると図 3 に示されるように、結果は予想された通りの直線回帰となることが

2　https://gpy.readthedocs.io/en/deploy/index.html
3　https://gpy.readthedocs.io/en/deploy/GPy.kern.src.html#module-GPy.kern.src.linear
4　線形カーネルがより単純に $k(x_i,x_j)=ax_i^Tx_j+c,\ a>0,\ c\geq 0$ と定義されている場合、variance は 1 つで $\sigma^2=a$ である。GPy.kern の定義は複数 variances に拡張されている。

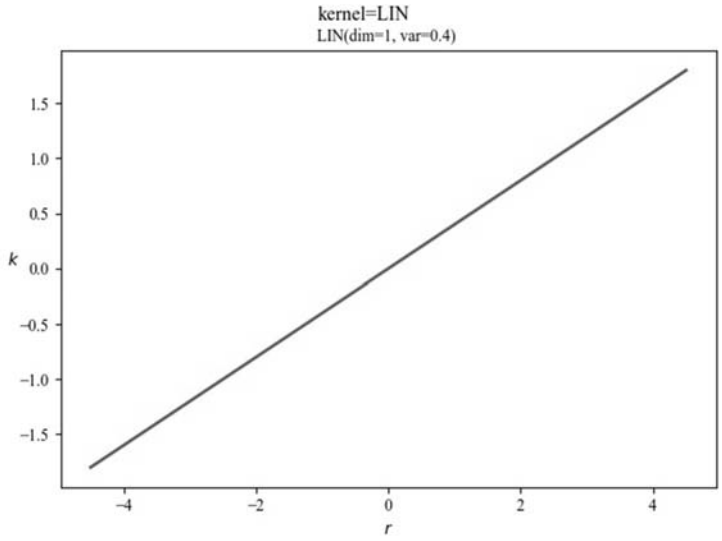

図 2　線形カーネル関数

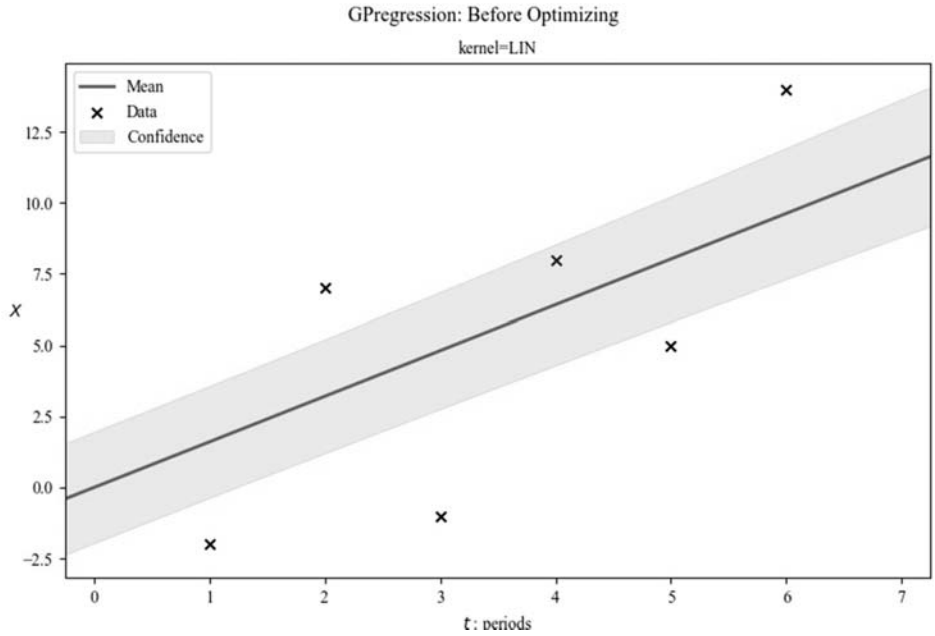

図 3　ガウス過程回帰（optimize 前）: 線形カーネル

分かる。

この結果は回帰として荒すぎるので改良したい。パラメーター（ハイパー・パラメーター: Hyper Parameters、σ^2）を調整することはせず、ここではモデル設計の見直しを行う。図3のデータは上方に反っているように見えるので次のモデルを考える。

（b）　線形カーネル×線形カーネル

線形カーネルを LIN と表すと、(9) から直ちに分かるように、LIN × LIN は r の 2 次式であるので、複合カーネルは、図4のような2次曲線となる。この複合カーネルを使ってガウス過

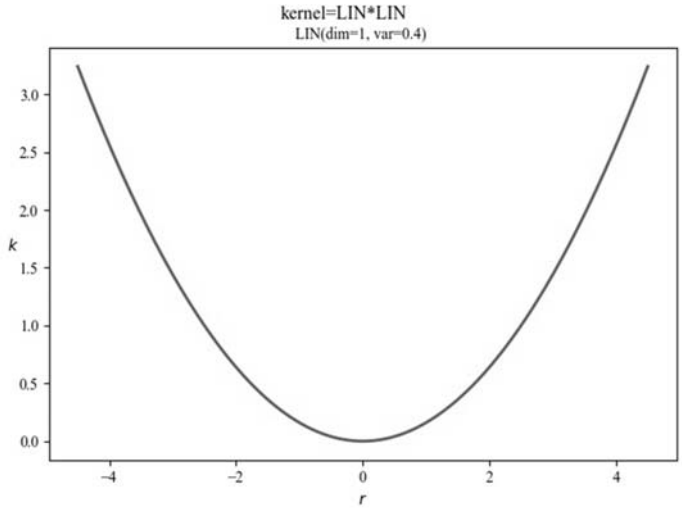

図 4　線形カーネル×線形カーネル

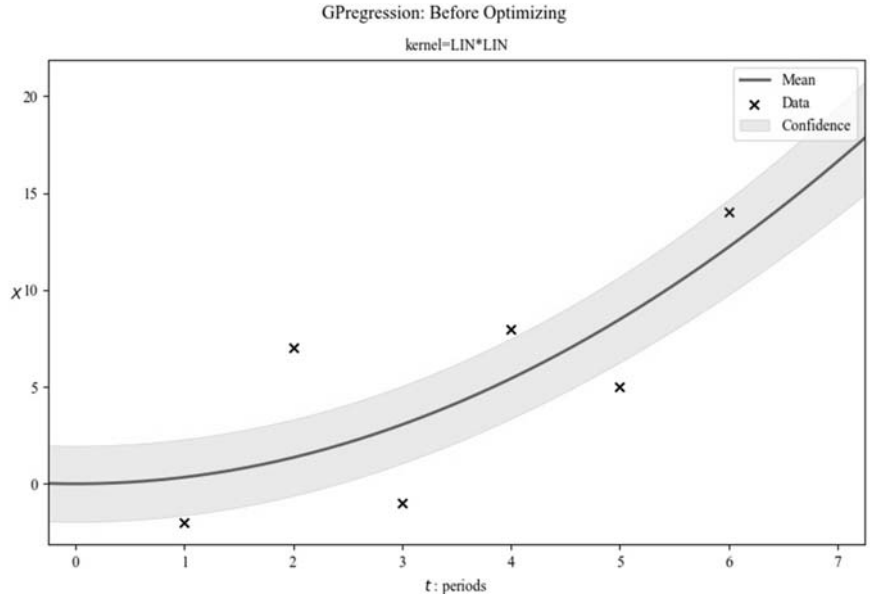

図 5　ガウス過程回帰（optimize 前）: 線形カーネル×線形カーネル

程回帰を行うと、予想どおり、図5に示したような2次曲線的な回帰曲線が求められる。

しかし図5を見ると、$t = 2$ の点のデータ（2, 7）が曲線から大きく外れていることが分かる。このような突出した変動に対する局所的な近似を考えると、次のように定常的変動を生み出すRBF関数を追加的に加えることが良いように思われる。

（c）　線形カーネル×線形カーネル＋RBF カーネル

図6は図4で示された変動なしの2次曲線に、局所的な変動が加えられたカーネル曲線を表している。この複合カーネルは、小変動をしつつも全体として2次曲線的なトレンドを示す回帰曲線を生むであろう、と予想される。実際ガウス過程回帰を行うと、図7のような回帰曲線と

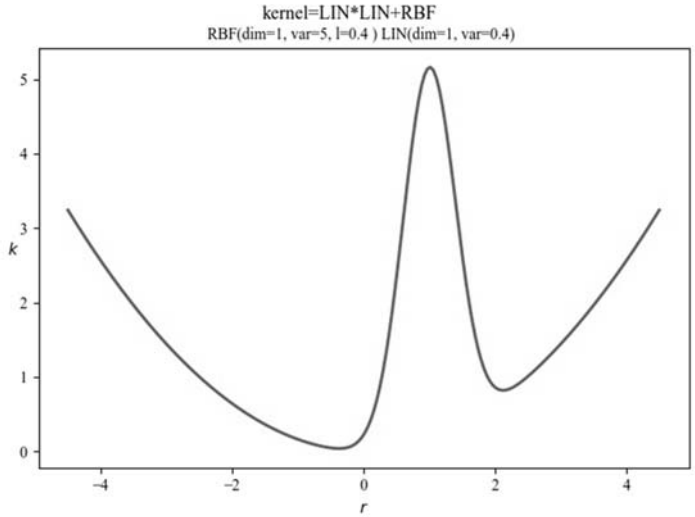

図 6　線形カーネル×線形カーネル＋RBF カーネル

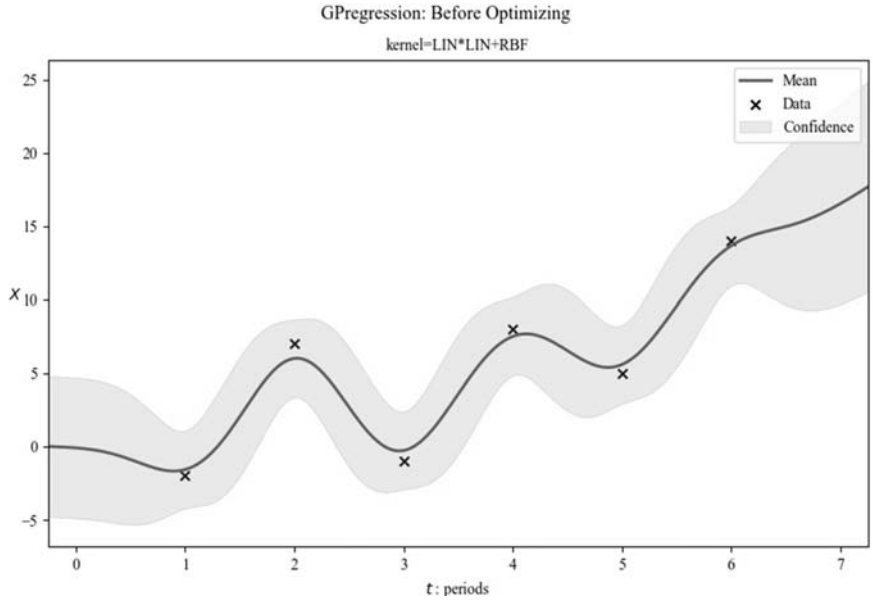

図 7　ガウス過程回帰（optimize 前）: 線形カーネル×線形カーネル＋RBF カーネル

なっていることが確認できる。

（iii）　定常的変動とトレンド変動に対応するカーネル

このようにデータの全体像を見渡した時、トレンドとして上がり下がりする部分とその周りを定常的に変動する部分とを分けることが可能であり、カーネル関数もそうした変動に対応して 2 つに分類することができる。以下では GPy.kern の中にある kernel モジュール群から実際に使って有用であると思われるものをいくつか取り上げる。

（a）　定常的変動に対応するカーネル

RBF カーネル

この関数は (5) 式で示されるが、Python プログラミングの中で使うには次のような形式で命

令する。ただし左辺の k は変数であって、knl でも ken でも何でもよい。

k = GPy.kern.RBF(*input_dim*, *variance*, *lengthscale*)

引数の *input_dim*（次元数）はデータに応じて次のように与える。時系列のように説明変数が 1 つの場合、訓練データは単なるベクトルであるから、次元数は 1 であり、*input_dim* には 1 を与える。消費が所得と価格と資産によって説明されるような計量モデルの場合訓練データは例えば次のような 3 × 5 行列になる。

	2011 年	2012 年	2013 年	2014 年	2015 年
所得	1025	1040	1035	1052	1047
価格	1.11	1.02	1.25	1.17	1.35
資産	5000	4890	5263	5300	5800

これの次元は縦と横で 2 である。従って *input_dim* には 2 が入る。仮にそのような行列が何層にも重なっているとするならば *input_dim* は 3 である。(4) 式の記法に従って表すと

$$\boldsymbol{x}_1 = \begin{bmatrix} 1025 \\ 1.11 \\ 5000 \end{bmatrix}, \boldsymbol{x}_2 = \begin{bmatrix} 1040 \\ 1.02 \\ 4890 \end{bmatrix}, \boldsymbol{x}_3 = \begin{bmatrix} 1035 \\ 1.25 \\ 5263 \end{bmatrix},$$
$$\boldsymbol{x}_4 = \begin{bmatrix} 1052 \\ 1.17 \\ 5300 \end{bmatrix}, \boldsymbol{x}_5 = \begin{bmatrix} 1047 \\ 1.35 \\ 5800 \end{bmatrix}$$

であり、$m = 3, n = 5$ で、共分散行列は $n \times n = 5 \times 5$ 行列となるが、次元数は 2 である[5]。

variance は変動の縦の大きさである。これが大きいといわば大波を生むイメージである。*lengthscale* は横の広がりを示し（(5) 式の λ に相当する）この値が小さいと尖った波になる。図 8 は *variance* = 10, *lengthscale* = 0.1 の RBF カーネルの形状を表しており、図 9 は先に示したデータ例からのガウス過程回帰を示している。この場合 GPy.kern の命令は

k = GPy.kern.RBF(1, 10, 0.1)

である。これから分かるように、元データを滑らかに回帰するか忠実に（over-fitting）回帰するかは *lengthscale* に大きく依存し、あまりに忠実に回帰しすぎると回帰の意味を失い、結

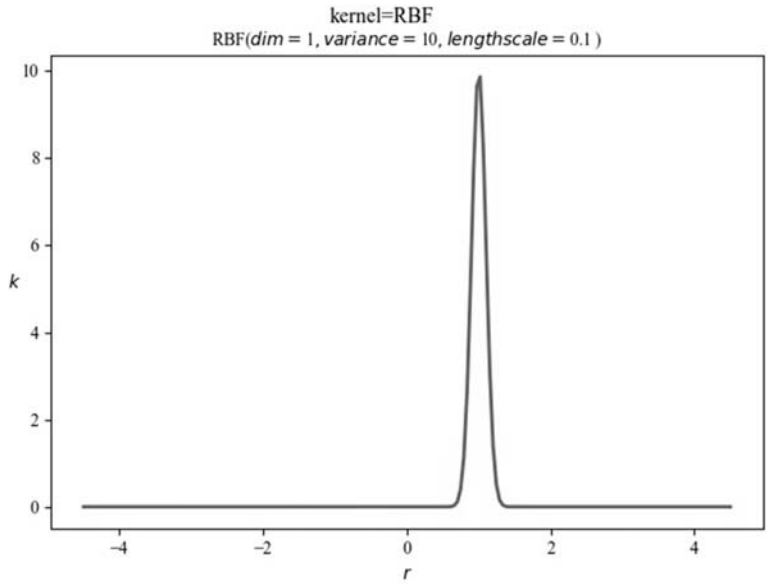

図 8　RBF 関数による尖ったカーネル

5　この次元数は numpy（http://www.numpy.org/）の ndim の考え方である。

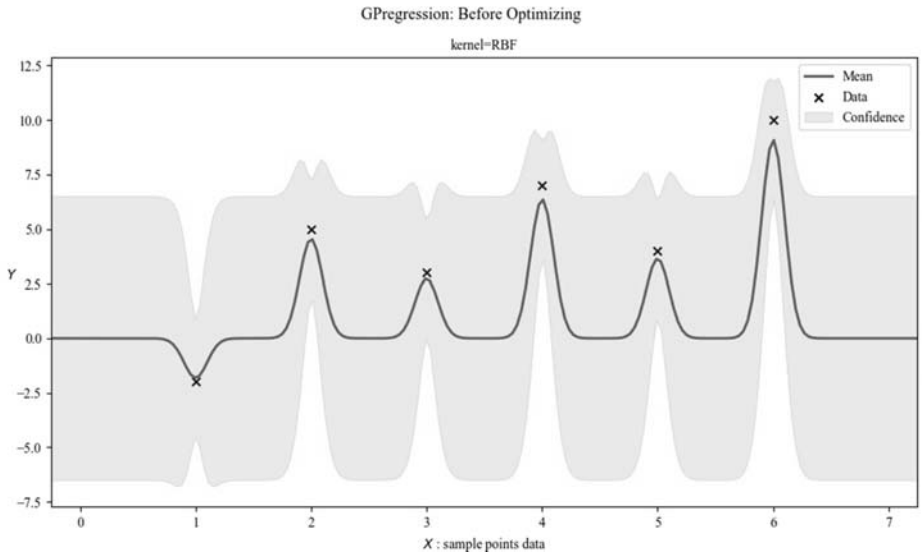

図 9　元データに追随しすぎた回帰

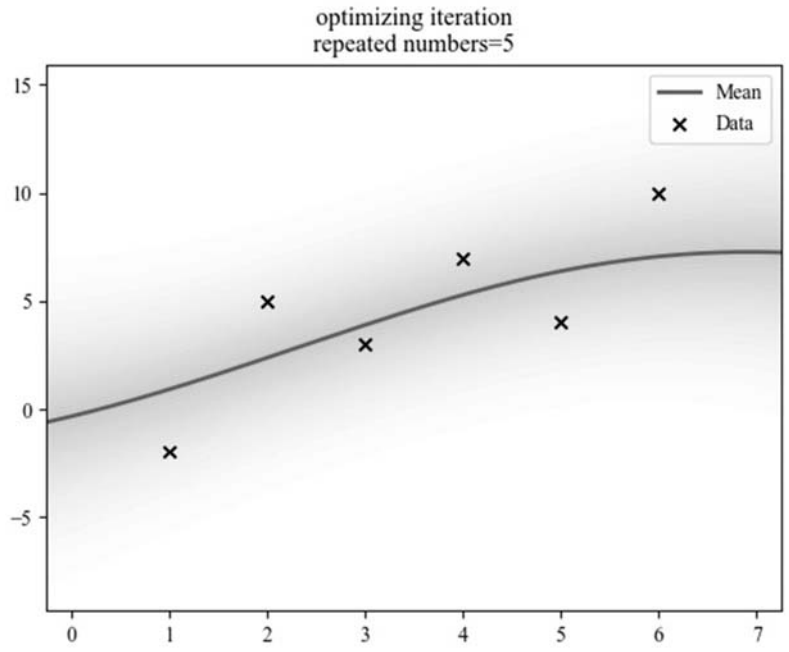

図 10　optimize による回帰の改善

果的に予測も不可能になる。さらに図 8 の場合は、元データがトレンド的に上昇しているにもかかわらず、定常的カーネルだけを用いて回帰したために、奇妙な回帰となったのである。ただし以上は初期設定であり、実際には再設定の optimize を行えば、optimeze 後の回帰が図 10 のように改善される。optimize はパラメータを変えて試行を行うので、何回か繰り返す必要がある。図 10 は 5 回 optimize を繰り返した後である。

Rational Quadratic カーネル

このカーネルは GPy.kern の Document の中では

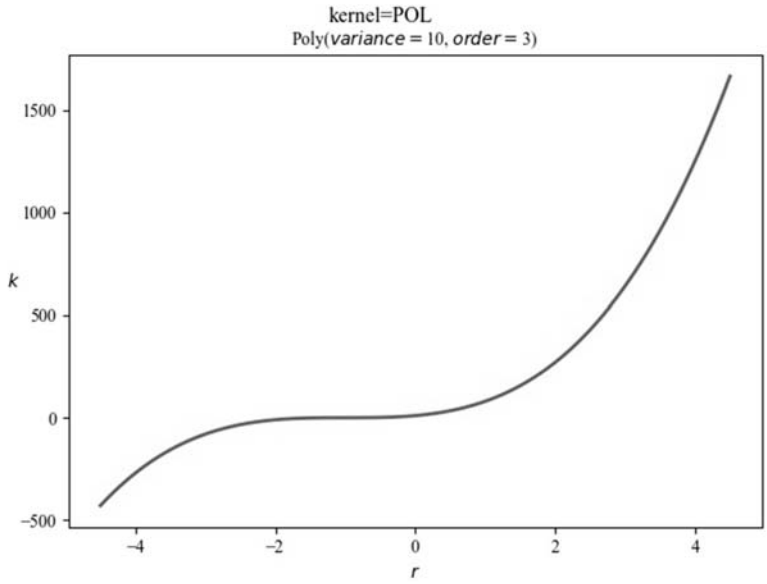

図 11　3次の Polynomial カーネル

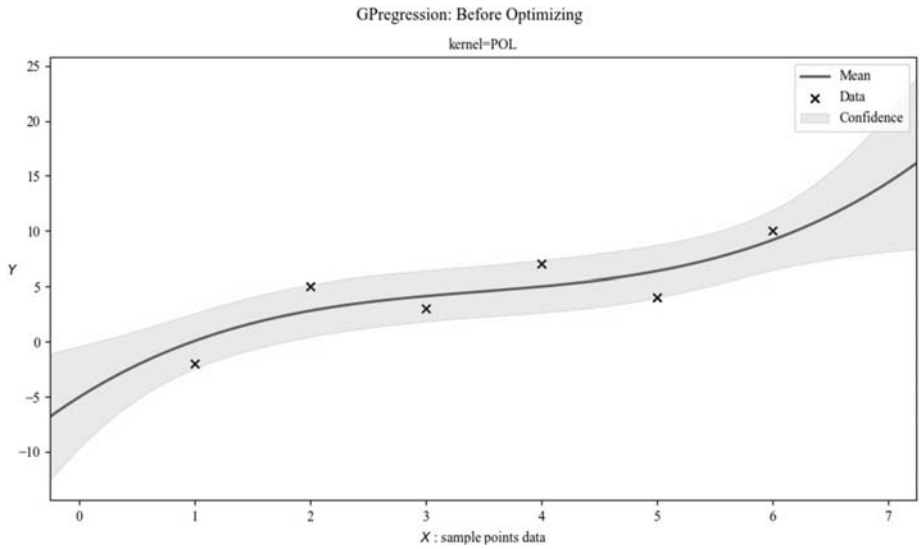

図 12　ガウス過程回帰（optimize 前）: Polynomial カーネル

$$k(r)=\sigma^2\left(1+\frac{r^2}{2}\right)^{-\alpha} \qquad (10)$$

と表されているカーネル関数で、σ^2 が *variance*、α が次の python 命令の中の power に相当し、この α が小さいほど変動は尖ってくる。変動の大きさのパラメーターと尖り具合のパラメーターの 2 つを持つので、RBF と同様に使いやすいカーネルである。Python プログラムの命令は

k = GPy.kern.RatQuad（*int_dim*, *variance*, *power*）

である。

（b）　トレンド変動に対応するカーネル

Polynomial カーネル

このカーネルは脚注2 で示された線形カーネ

ル関数 $ax_i^T x_j + c$ を l 乗した

$$k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \left(a\boldsymbol{x}_i^T \boldsymbol{x}_j + c\right)^l$$

で表される。GPy.kern プログラムの命令では

k = GPy.kern.Poly(*input_dim*, *variance*, *scale*, *bias*, *order*)

であるが、*scale* と *bias* はデフォルトで 1.0 となっているので例えば次元数が 1 で *variance* が 10、l = *order* = 3 の場合、python 命令を

k = GPy.kern.Poly (1, 10, 1.0, 1.0, 3)

として実行するとカーネル曲線は図 11 のような 3 次曲線となり、予想通り optimize 前のガウス過程回帰は図 12 のような 3 次曲線的なトレンドで回帰されている。

Ⅱ　時系列データを用いたシミュレーション

(i)　多項式を用いた parametric 予測の概要

本稿の主目的は著者［7］,［8］の parametric 予測法（Benigma と名付けた）と non-parametric 予測法であるガウス過程回帰予測との比較検討にあるので、まず前者の概要を示しておく。

第 1 によく見られる多項式回帰と異なる点は、N 次多項式にさらに少数次（$0 \le r < 1$）が加わっている点である。例えば次のような回帰式ならば 3.62 次式と呼ぶことにする。

$$Y = a_0 + a_1 t + a_2 t^2 + a_3 t^3 + a_4 t^{0.62} \qquad (11)$$

第 2 にこのような次元を先験的に決めて回帰するのではなく、次元を小刻みに増やしていって最適次数を探索する。その際の最適性の評価はデータ最終点から前の例えば 10 期間の評価期間の平均絶対値誤差率平均を使う。図 13 は 100 日分の株価終値データ [6] をもとに $r = 0.1$ から始めて 0.05 ずつ次元を上げていった時の「10 期間平均絶対値誤差率」の推移を表しており、次元がちょうど 4.0 の時に大域的な最低点になっていることが分かる。図 14 はこの次数で回帰曲線を 20 期延長した時の予想が示されている。

図から読み取れるように、この方法は予測期

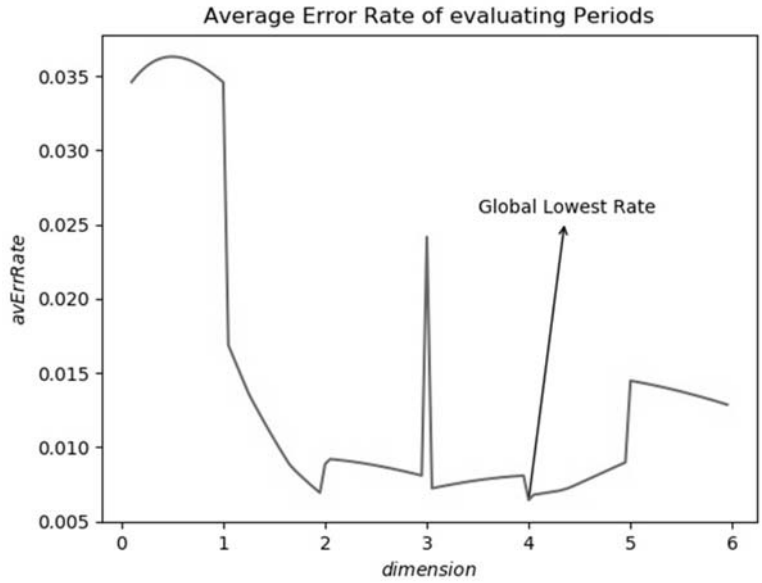

図 13　回帰次数に伴う評価期間（直近 10 期間）の平均絶対値誤差率の推移

6　日立製作所 2015/1/5 から 100 日分を使用。データは全く無作為的に選択した。また over-fitting を避けるため最大次数は 8 次に設定してある。

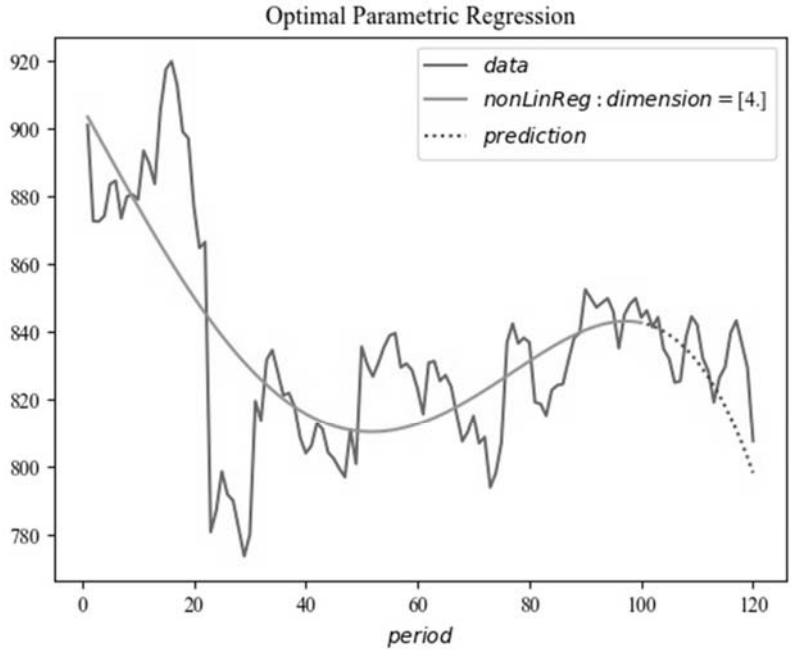

図 14　多項式 parametric 回帰による予測

間がかなり長くなると、現実の値から急速に離れていく。多項式 parametric 予測は特に多項式の次数が高いほうの係数が大きいと、時間 t が大きくなるにつれ曲線の傾きが逓増（または逓減）し、極端な方向へと発散して収拾がつかなくなるという多項式回帰特有の欠点がある。

(ii)　ガウス過程回帰による予測

ガウス過程回帰は前述のようにカーネル関数の持つ特徴を色濃く反映する。しかしそれを利用して、パラメトリック予測の持つ弱点を抑制することが可能である。例えば株価や為替相場のように時系列が一方的にどこまでも発散することが少なく、いわゆる「平均回帰的（mean reverting）」な性質を持つ場合は、RBF や Rational Quadratic（RTQ）のような定常的カーネルを組み込むことで穏やかな予測を生み出すことが可能である。非定常的なカーネルである Polynomial カーネル（*variance* = 3, *order* = 3 を初期値とする）単独でガウス回帰予測を行うと図 15 に示すように、現実値を大きく外れた発散的予測となってしまう（図中の Non-Param（Greg）曲線）。従ってこのカーネル単独では parametric 回帰と何ら変わりない。しかし、このカーネルに定常的カーネルである RTQ を加えて

$$\mathrm{POL} + \mathrm{RTQ}$$

のようなモデル設計を行うと、RTQ の平均回帰性を反映して、図 16 のような、発散が抑えられた予測が得られることが分かる。

図 16 は多項式 parametric 予測とガウス過程回帰予測の間の関係に重要なヒントを与える。多項式 parametric 予測は予測の方向性に関しては精度がよいが、しかし先に進みすぎると極端に発散的となる。一方ガウス過程回帰予測は適当な平均回帰性カーネルを組み込めばそうした極端な発散を抑制することが可能である。しかし図 16 では、両者が現実の変動の上端と下端を予測し、その間の領域を通っていないという意味で、発散と過剰抑制の持つ 2 つの欠点を図らずも表す形となっている。すると次の予想が考えられる。より精度のよい予測は両者の加重平均なのではないか。

例えば両者に同等のウェイト 0.5 を与えたグラフが図 17 である。図 17 の一点鎖線で示され

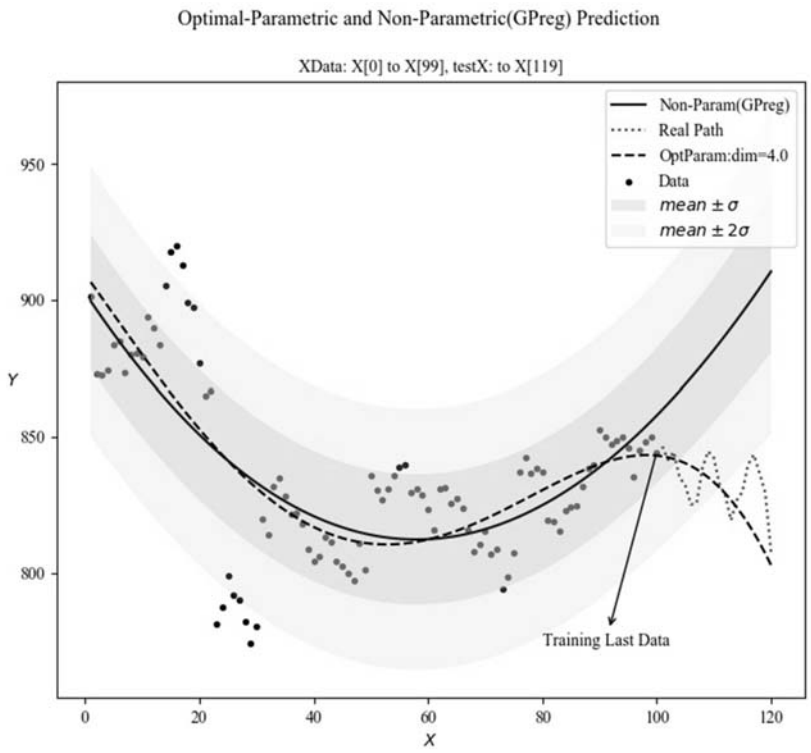

図 15　Polynomial カーネル単独によるガウス過程回帰予測

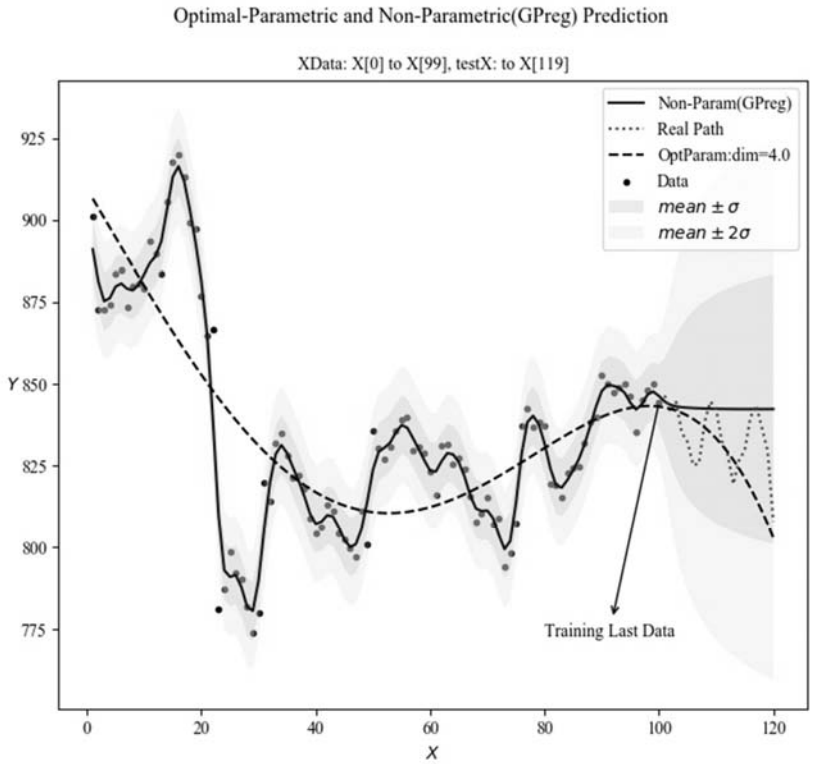

図 16　POL+RTQ カーネルによるガウス過程回帰予測

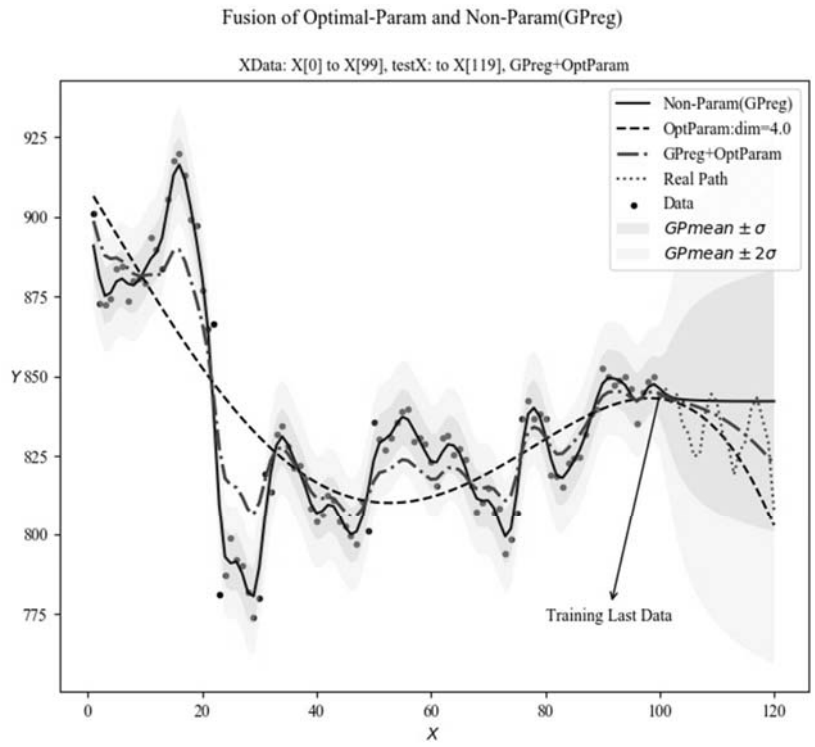

図 17　ガウス過程回帰予測と Optimal Parametric 予測の融合（GPreg+OptParam）

た曲線（Gpreg+OptParm）が

0.5 × GP 回帰予測 + 0.5 × 多項式parametric 予測

を表す曲線であり、20期間現実値のほぼ真ん中を横切っていることが読み取れる。平均誤差率誤差率で測っても次のように融合的予測の方が精度において改善されていることが分かる。

	20期間誤差率絶対値平均（%）
ガウス過程回帰予測	1.187186014
多項式parametric 予測	1.147756921
0.5*GP 予測 +0.5*OptParam 予測	0.926965193

Ⅲ　日経平均データによる検証

以上のようにガウス過程回帰予測は、カーネルをうまく設計すれば抑制的な予測によりparametric 予測の極端な発散性を中和する働きを持つことが分かったが、相場の地合いが突然変化するような場合は、変化の速さに予測が追随できないという欠点を持つかもしれない。このことを日経平均データ（2017年10月2日から）で検証してみよう。

まず10月2日から100日のデータを使い、101日目（2018年3月1日）から先20日分の予測を行う。ただし多項式parametric 予測の誤差率評価期間は5日と設定し、ガウス過程回帰のカーネルは RTQ + POL とし、POL の *order* は前節と同じく 3 と初期設定する。この場合相場は基本的に下がり気味で、地合いの変化はなく、図18 に示すように前節の事例とちょうど逆転する形ではあるが多項式parametric 予測とガウス過程回帰予測が現実の時系列を挟み込んでいる。従って最良の予測はその中庸をとるのが良いと考えられ、実際図中の Gpreg + OptParam 曲線は良好な予測曲線となっていることが分かる。

次に 2017年10月30日から 100日分のデータを使い 101日（2018年3月29日）から先20日分の予測を行う。相場は次の日から突然急上昇に転じ、全く別の様相を見せ始める。両予測ともに図19 に示すように、当然この変化を予測で

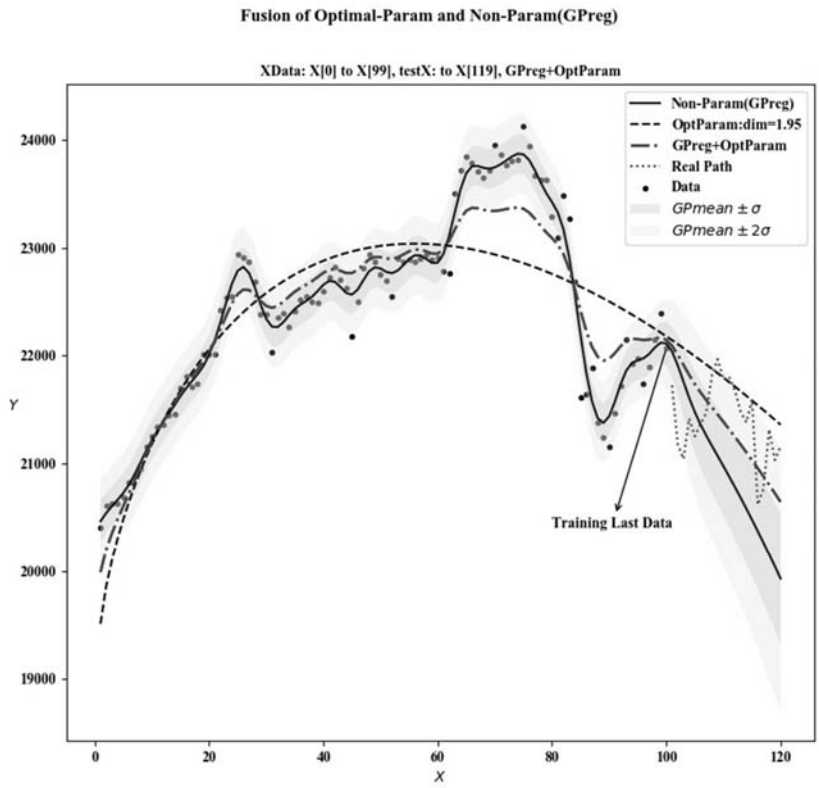

図 18　日経平均予測: 地合いの変化がない場合

きない。そこで問題は、データの急激な変化に対して予測をいかに早く修正できるかということである。取り込むデータと予測をずらして予測の修正の様子を観察してみよう。図20 は変化から 2 日後の 20 日予測である。ただし 100 日分の訓練データも 2 日分ずらしてある。図から読み取れるように、多項式parametric 予測もガウス過程回帰予測も 2 日経てば完全ではないが半ば予測の修正に成功している。計算すると前者の 20 日予測誤差率平均は 1.769%、後者は 1.734% でほぼ同じである。ただしガウス過程回帰予測の方が、後半に「中心回帰性」の性質が災いしてその経路が実データから外れ始めているのが分かる。図21 は変化から 3 日後の 20 日予測である。多項式parametric 予測はすでに完全に予測の修正に成功しているのに対し、ガウス過程回帰予測はやはり後半の予測に難がある。20 日平均をとると次のようになる。

	20 期誤差率 絶対値平均 (%)
ガウス過程回帰予測	1.157046741
多項式 parametric 予測	0.827999598
0.5*GP 予測 +0.5*OptParam 予測	0.642993331

結果は予想通り、両予測の加重平均をとるのが最良である。ただし図20、図21 を詳細に見ると、予測の前半はガウス過程回帰予測がより一層現実値に粘着的に絡みついているのが分かる。これはガウス過程回帰の定常的カーネルの働きによるものであると考えられる。それに対して多項式parametric 予測は小変動を慣らしたトレンド的回帰という性格により後半に強い。ただし前述のように、多項式parametric 予測はさらに先に行くと収取がつかないほど現実値から乖離・発散するであろう。紙面の関係で詳細は省略するが、試してみると、以上の性質は日経平均データをはじめとする様々な株式価格の時系列に対して共通に当てはまる。以上から次の結論を導くことができる。

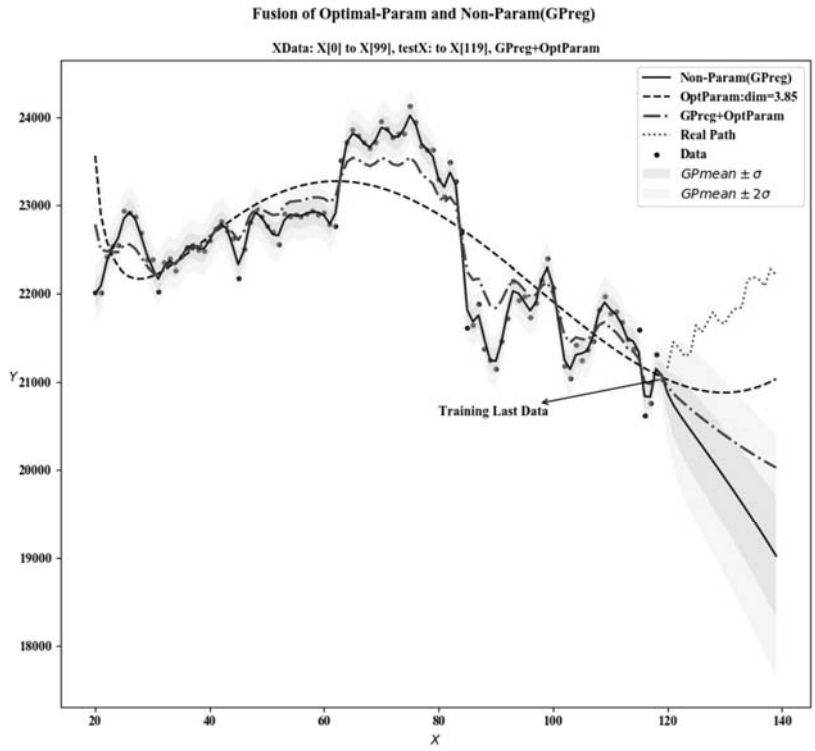

図 19　突然の地合いの変化により予測に失敗する場合

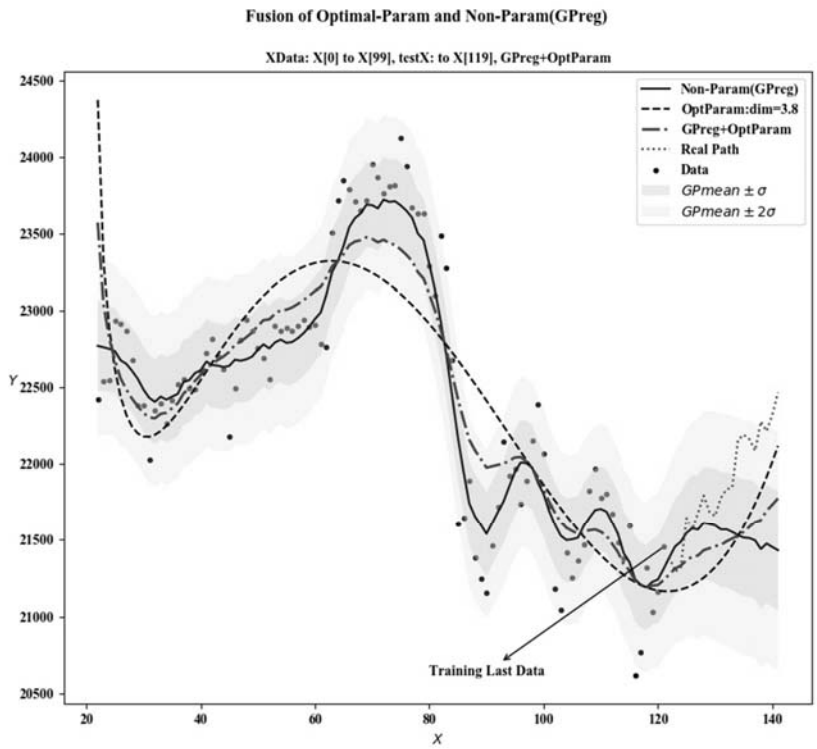

図 20　変化から 2 日後の予測修正

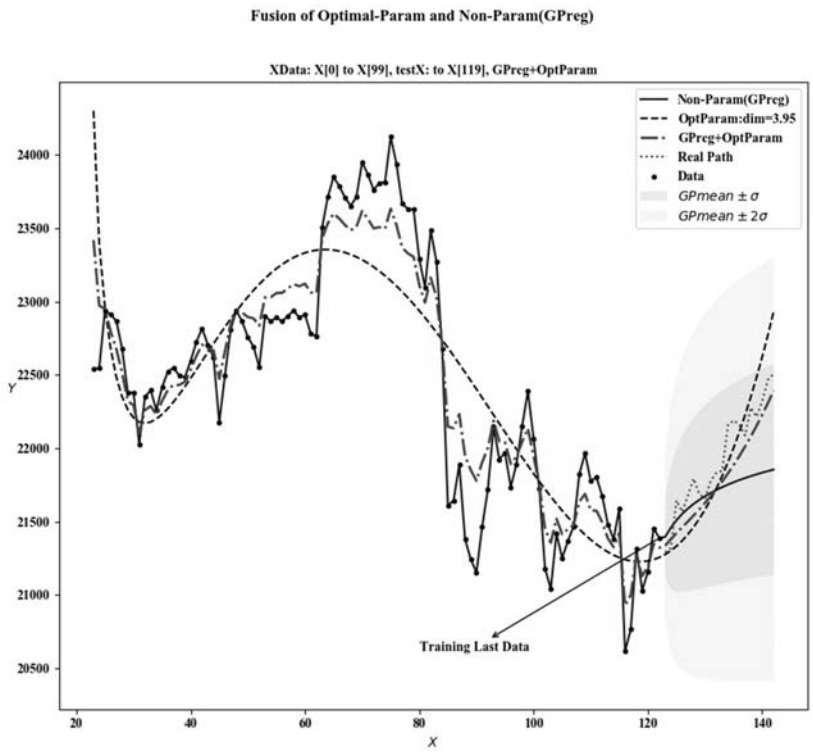

図 21　変化から 3 日後の予測の修正

Ⅳ　結論

(1) ガウス過程回帰予測にはカーネルの組み合わせ（モデル設計）が不可欠である。

(2) モデル設計を「定常的カーネル＋非定常的カーネル」とすることにより小変動も大変動（トレンド）も捉えた予測が可能となる。

(3) 定常的カーネルの働きにより、短期的（日経平均の場合10日前後）変動に対して極めて精度の良い予測が可能となる。特に相場の急変に対して迅速な修正（2日程度のタイム・ラグ）が行われる。

(4) ただしその抑制的･平均回帰的性格により、長期変動に対する予測力は相対的に弱いかもしれない。

(5) 同期間で比べると長期的予測力は多項式parametric予測の方が相対的に強いが、発散的性質から逃れることができないのでそれ単独では現実値から乖離する危険性がある。しかしガウス過程回帰予測と加重平均することにより両者の強みが互いに補強され、平均的により精度の高い予測が可能となると期待される。

(6) 従って結論として両者をウェイトで加重した融合的な予測方法が良いということになる。ただしウェイトを 0.5 としてよいかは今のところ確証はない。

なお著者は［7］以来平均誤差率1%以内の予測が可能であるか模索していた。なぜ1%なのかというと、為替相場では、1ドル100円とすると1%誤差では1円の誤差になってしまうからである。これでは実用的ではない。ところが［7］でBenigmaと名付けた多項式parametric予測と、本稿で検討したガウス過程回帰予測を加重平均すると、上の表に示したように確かに日経平均データで20日平均0.64%誤差率が実現した。日経平均と為替相場は誤差率を求める際の分母が異なるので､単純な比較はできないが、これは大きな前進であると考える。ガウス過程回帰にさらに学習効果の強いAI的手法や記憶（メモリー）効果などを組み込めばさらに平均誤差率は下がると期待される。

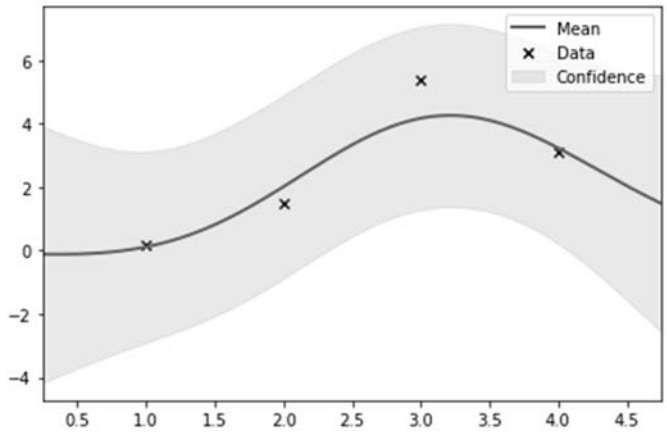

図 22　Spyder の IPython コンソールに出力されたガウス過程回帰

Ⅴ　（補論）GPy の使い方

GPy は Sheffield 大学のコンピュータ・サイエンス学部の機械学習グループから提供されているガウス過程関連の、python で書かれたソフトウェア群である。ベイズ最適化を行う GPyOpt とともにガウス過程回帰予測に最適なモジュールがこの中に用意されている。使用の前提として Python, numpy, matplotlib がインストールされていなければならない。ただしこれらは Anaconda（https://www.anaconda.com/）をインストールすれば、付随的にインストールされる。Anaconda に付属する Spyder などの開発環境を使い必要なツールを次のように import する。

```
import GPy
import numpy as np
from matplotlib import pyplot as plt
```

データは numpy の次元数 2 の行列（またはベクトル）で与える必要がある。例えば

```
X=np.array([[1],[2],[3],[4]])
Y=np.array([[0.2],[1.5],[5.4],[3.1]])
```

のようにである。これらは通常の縦ベクトル表示だと

$$\boldsymbol{X}=\begin{bmatrix}1\\2\\3\\4\end{bmatrix},\quad \boldsymbol{Y}=\begin{bmatrix}0.2\\1.5\\5.4\\3.1\end{bmatrix}$$

に等しい。あるいは Excel の表では

	A	B
1	X	Y
2	1	0.2
3	2	1.5
4	3	5.4
5	4	3.1

である。Excel 上に入手した多数のデータをいかに numpy 形式の配列に書き換えるかという問題は GPy を利用する際の最大の問題である。著者は VBA でこの作業を自動化しているが、ネットを検索すると変換アプリケーションが利用できるようである。

道具立てはこれだけで、基本的に命令文には kernel を定義する文と GPRegression を命じる文とグラフを書く plot 文が最低必要である。まずカーネルを

```
k=GPy.kern.RBF(1,1,1)
```

のように定義する。これを使い GPRegression を

```
model=GPy.models.GPRegression(X,Y,k)
```

のように命令する。しかしこのままでは最適化されていないので、optimize を 3 回ほど反復させる。for 文を使うまでもなく

```
model.optimize ()
```

を 3 回繰り返し、最後にその結果を次のようにプロットする。

```
model.plot ()
plt.show ()
```

以上をまとめると spyder の中に次のように書き込んで「実行」すれば結果は図 22 のように Spyder の IPython コンソールに出力される。

```
import GPy
import numpy as np
from matplotlib import pyplot as plt
X=np.array ([[1],[2],[3],[4]])
Y=np.array ([[0.2],[1.5],[5.4],[3.1]])
k=GPy.kern.RBF (1,1,1)
model=GPy.models.GPRegression (X,Y,k)
model.optimize ()
model.optimize ()
model.plot ()
plt.show ()
```

［参考文献］

[1] Bishop, Christopher M., *Pattern Recognition and Machine Learning*, Springer, 2006（『パターン認識と機械学習（ベイズ理論による統計的予測）上下 元田浩ほか監訳　丸善出版　2012』）.

[2] Duvenaud, David Kristjanson, *Automatic Model Construction with Gaussian Processes*（dissertation for Doctor of Philosophy, University of Cambridge）, https://www.cs.toronto.edu/~duvenaud/thesis.pdf, 2014.

[3] GPy: https://gpy.readthedocs.io/en/deploy/

[4] GPy kernel: https://gpy.readthedocs.io/en/deploy/GPy.kern.html

[5] Rasmussen, Carl Edward and Williams, Christopher K. I., *Gaussian Processes for Machine Learning*, The MIT Press, 2006.

[6] Shawe-Taylor, John and Cristianini, Nello, *Kernel methods for Pattern analysis*, Cambridge University Press, 2004（『カーネル法によるパターン解析』大北剛訳　共立出版　2010）.

[7] 荒木勝啓「非線形時系列予測の最適化」駒澤大学経済学論集 第44巻　第3号2016, pp.33-46.

[8] 荒木勝啓「日経平均への最適化非線形予測の適用」駒澤大学経済学論集 第44巻　第4号2016, pp.35-48.

[9] 荒木勝啓「AI手法（ガウス過程）を用いた予測−理論編」駒澤大学経済学論集 第50巻 第2号 2019, pp.1-14.